(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 479 520 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **91308919.9**

(22) Date of filing : **30.09.91**

(51) Int. Cl.^{5} : **G06F 15/16,** G06F 13/12, G06F 13/42

(30) Priority : **05.10.90 US 593824**

(43) Date of publication of application : **08.04.92 Bulletin 92/15**

(84) Designated Contracting States : **DE FR GB**

(71) Applicant : **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504 (US)

(72) Inventor : **Gould, Joel Mark**
24 Prescott Road
Norwood, Massachusetts 02062 (US)

(74) Representative : **Moss, Robert Douglas**
IBM United Kingdom Limited Intellectual Property Department Hursley Park
Winchester Hampshire SO21 2JN (GB)

(54) **Network communication apparatus.**

(57) Apparatus and method for transmitting control information in a multidimensional computer network are disclosed herein. A multidimensional computer network consists of a number of nodes (1,2) that are able to receive data blocks from other nodes and send data blocks to other nodes. Interconnecting the nodes are links (12, 13) which connect the nodes in a regular or irregular pattern. Bandwidth is stolen from the data transmission facilities on demand without requiring prior reservation of a fixed amount of bandwidth and is used to transmit control information in small data transmissions.

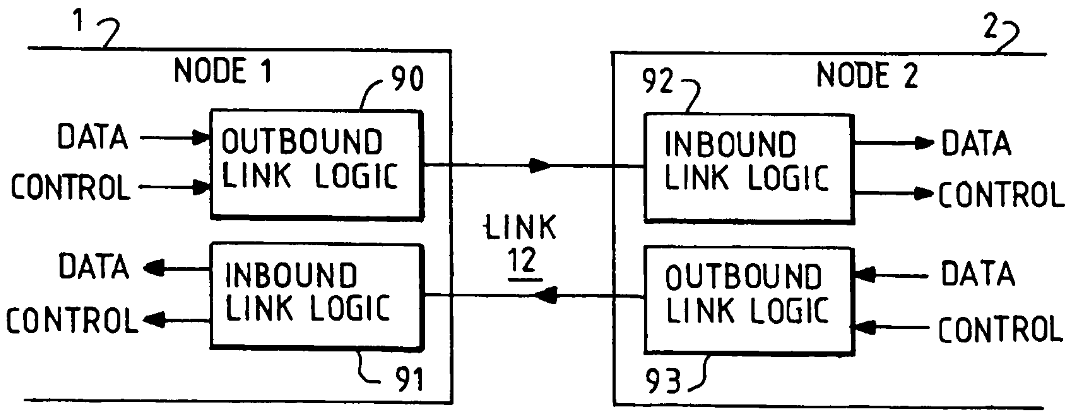

FIG. 2

Jouve, 18, rue Saint-Denis, 75001 PARIS

This invention generally relates to communication systems comprising a plurality of nodes, and more particularly to a technique that enhances management of a multidimensional computer network.

Since the advent of the electronic computer in reliable form, workers in the art have given much consideration to systems employing a number of computers functioning together in interrelated fashion to accomplish a given overall task. In some of these multiprocessor systems a large computer utilizes its superior speed and capacity to perform the complex portions of a program, but assigns smaller and slower satellite processors the less complicated and less urgent tasks in order to reduce the load and demands upon the large computer. The large computer is required to undertake the responsibilities of assigning subtasks, making sure that the smaller processors are kept occupied, ascertaining the availability and performance of the smaller processors, and providing a unified result.

Other multiprocessor systems utilize a different approach, employing a plurality of processors and a common bus system, with the processors having essential equality of function. In this type of system, separate control computers or control systems are often used to monitor the availability and capability of an individual processor for a given subtask, and to control the routing of tasks and information between processors. The processors may be arranged and operated so that they monitor the status and availability of the other processors and determine the routing of messages and programs. The common and substantial drawback of these systems is that the software and operating time required for overhead and maintenance functions interfere with the performance of the principal objectives. Problems of routing and monitoring may increase dramatically in relation to the number of processors involved, so that ultimately a disproportionate amount of effort is spent in overhead functions.

Since the days of the early "Binac" (two parallel processors) aid comparable systems it has been recognized that a multiprocessor provides a redundant capability that can substantially improve the overall reliability of an operating system. Actual installations of multiprocessor systems have until recently been quite limited, largely due to the extensive software problems involved. Nevertheless, the advantages of multiprocessor operation for real time applications and other situations in which system down time cannot be tolerated have led to the development of systems which are successful in operation but which nevertheless involve significant commitments to overhead software and operating time.

Illustrative of multitasking systems are U.S. Patents 3,445,822, 3,566,363 and 3,593,300, all relating to a system in which multiple computers access a single shared main memory, and in which capabilities and requirements are compared in order to assign tasks optimally to individual processors.

Another example of the prior art is U.S. Pat. No. 4,099,233, in which a number of processors share a single bus and a control unit incorporating a buffer register is used in the transfer of data blocks between a transmitting
miniprocessor and a receiving miniprocessor. This concept has been employed in a distributed mail sorting system in Europe.

U.S. Patent 4,228,496 pertains to a multiprocessor system in which buses between processors are coupled to bus controllers which monitor transmissions and determine the priority of data transfers between processors, each of which can be coupled in to control a certain part of a number of peripheral devices.

The "Ethernet" system, described in U.S. Patent 4,063,220 and 4,099,024 discloses another approach to the problem of intercommunicating between different processors and peripherals. All units are coupled to a common multiple access network and compete for priority. Collision detection is based upon time priority, which in turn means that global capabilities cannot readily be controlled, coordinated or given specificity.

Details of these complex systems can only be fully appreciated by analysis of the patents and any related publications. However, review will verify that in each instance prioritizing of data transfer and the selection of processors requires extensive intercommunication and supervisory control if tasks are to be shared. Expansion of the systems to include additional processors does not present identical problems with these different systems, but in each instance substantially complicates system software, applications programming, hardware, or all three.

In most networks, each message is positively acknowledged when it reaches the destination node by sending an acknowledgement word back along the path used to transmit the message (this technique applies to both circuit switched and packet switched networks). Thus, a source node enqueues its message into the network and starts a countdown timer. If the positive acknowledgement is received before the timer expires, the message transmission is considered successful. If the timer expires before an acknowledgement is received, the message transmission is considered a failure and the message must be resent.

The problem with the above scheme is that the source node only knows if the message succeeds or fails, no maximum progress report is given. Message failure might have been caused by several reasons including network congestion, broken links, transmission errors, and node errors. Since the source node does not know the cause of the failure or the maximum message progress, its only option is to retry sending the message or abort the message.

An obvious improvement to the above scheme is to add negative acknowledgements that indicate why the message failed. Thus, some single-dimensional (one path) networks return specific status words when the message transfer has failed. However, this method does not scale to multidimensional networks where multiple paths can fail. In the above example, suppose Node 1's children, Nodes 2 and 4, both fail sending the message for different reasons. They both return this information to Node 1. Node 1 must then return both errors to Node 0, requiring Node 0 to process every error that occurs on every search path. While this protocol satisfies the expendability need, it requires too much overhead memory by not encapsulating status information into a single status word. Furthermore, too much network bandwidth is required to support this method.

A final refinement to the scheme would allow intermediate nodes to accumulate error reports and pass only the most severe error report to its parent. Although this scheme reduces the acknowledgement message traffic, it does not encapsulate the message progress through the network. Using this scheme, the source node is only informed of the most severe error occurring on any search branch without any information regarding message progress being reported.

Inherent constraints on multiprocessor system size and capability are imposed by the use of one or two logically passive busses. While different techniques can be employed to facilitate intercommunication, such as the grouping of subsystems into global resources evidenced in U.S. Pat. No. 4,240,143, the amount of useful traffic must reach a limit and variable delays impose insuperable problems when large numbers of processors are used. Situations may arise in which one or more processors become locked out or deadlocked, and these circumstances in turn require added circuitry and software to resolve the problems. The impracticality of substantially extending the number of processors, say to 1024, thus becomes evident.

It is desirable for many applications to depart from the constraints of these existing approaches and to utilize modern technology to best advantage. The lowest cost technology available today is based upon mass produced microprocessors, and high capacity rotating disk memories, such as Winchester technology devices using small head to disk spacing in a sealed environment.

It is desirable to expand a multiprocessor system without disproportionate or even concomitant software complexity. It is desirable further to be able to handle computer problems that may be characterized as having a distributed structure, in which an overall function can be dynamically subdivided into limited or iterative processing tasks.

Virtually all data base machines fall into this category, which also includes such other typical examples as sorting, pattern recognition and correlation, digital filtering, large matrix computations, simulation of physical systems and the like. In all of these situations there is a requirement for widely dispersed, relatively straight-forward individual processing tasks with a high instantaneous task load. This situation unduly burdens prior art multiprocessor systems because it tends to increase the time and software involved in overhead, and because practical difficulties arise in implementation of the systems. Using a shared passive bus, for example, propagation rates and data transfer times introduce an absolute barrier as to the rate at which transactions can be processed.

This invention addresses the problem of transmitting control information in a multidimensional computer network. A multidimensional computer network consists of a plurality of homogeneous or heterogeneous processing units, called nodes. All that is required of a node is that it be able to receive data blocks from other nodes and send data blocks to other nodes. A node, for example, could be a personal computer comprising a display, disk, communication adapter and keyboard. Interconnecting the nodes are links which connect the nodes in a regular or irregular pattern as shown in Figure 1. A nodal link could be as simple as an asynchronous communication cable connecting two personal computers. Alternatively, a local area network could provide multiple communication links for each node as shown in Figure 1. To manage nodal information exchange in a multidimensional computer network, it is necessary for nodes to exchange control information with every transmission of information.

Accordingly, the invention provides network communication apparatus for transmitting selected control information and message data between a plurality of nodes, comprising:

(a) communication means for transmitting message data;

(b) memory means for storing portions of the message data;

(c) processor means for selecting appropriate control information to transmit based on current communication status; and

(d) multiplexor means for transferring selected control information or message data to the communication means for transmission to a subsequent node.

The invention facilitates the transmission of control information in addition to message information between nodes on a multi-node system without reserving bandwidth and without impacting other messages sent via the network.

Typically, a multi-nodal, computer system employs a set of high speed nodal links to network messages between a plurality of nodes. Each node and its associated nodal link has communication transmittal means for routing control information between nodes without impacting message transmission on the network. In a preferred embodiment, each nodal link is a full duplex communication link. Thus, whenever a message is transmitted in one direction and no corresponding incoming message is present, the full bandwidth of the link can be employed to transmit control information in the opposite direction without impacting communication system performance. The invention exploits this capability to transmit control information associated with information exchanged between nodes.

On occasion, transmission of data messages in both directions may be active when the control information should be sent. When this is the case, the data message is temporarily stored in a buffer while the control information is transmitted. If too many messages are incoming to a particular node, and the buffering means are insufficient to store all messages, then a control block is transmitted back to the node sending the data messages to temporarily block message transmission to the node.

The subject invention is applicable to links of any length and speed and to communication networks of any size. The network may interconnect tightly coupled processors in a parallel computer or heterogeneous processors distributed over a wide geographic area. The invention applies to multidimensional, or parallel networks as well as single dimensional networks where only one path exists between any two nodes.

Bandwidth is stolen from the data transmission facilities on demand without requiring prior reservation of a fixed amount of bandwidth. The bandwidth is used to transmit control information in small data transmissions without impacting message transmission on the network.

The invention also provides a method for communication in a network comprising a plurality of nodes, the nodes comprising processor means, memory means, multiplexor means and communication means, whereby selected control information and message data is transmitted between nodes, said method comprising the steps of:

(a) storing a portion of message data in said memory means;

(b) selecting appropriate control information to transmit based on current communication status; and

(c) transferring selected control information or message data via the multiplexor means to the communication means for transmission to a subsequent node.

The easiest way to design a network in which messages are mixed with control information is to reserve bandwidth for the transmission of control information. For example, if each link comprises eight wires for transmitting and receiving data in each direction, seven of those wires could be used to send messages and one of those wires could be reserved for control information. In such a scheme, there would always be one wire available for sending control information even if a message was being sent over the link.

The problem with this scheme is that in the absence of control information, only seven eighths of the bandwidth of the link would be available for sending messages. The eighth wire, which is reserved for control information, would remain idle and its bandwidth would be wasted when control information was not transferred.

Using time-multiplexing has the same problem. For example, a communication system employing every tenth byte sent over each link for control information is feasible. However, in such a system, only nine tenths of the available bandwidth would be utilized in the absence of control information. Likewise, if certain bytes of every message were reserved to facilitate transmission of control information, that bandwidth would be wasted in the absence of control information at the time that those bytes were sent.

If bandwidth cannot be reserved for control information, then the network must be designed to allow control information to temporarily interrupt the transfer of a message across that same link. One common way to implement this type of network is to split every message into fixed size blocks, called packets, which are sent in order through the network. When control information needs to be inserted, the link can complete the current packet, buffer the next packet to be sent and send out control information instead of the packet.

In its simplest implementation, the packet which is replaced by backwards control information can be buffered until the end of the message. However, this implementation requires that every intermediate node have at least enough memory for one packet. It also implies that the message may arrive out of order at the destination node which is a severe limitation.

To avoid sending data out of order, the displaced packet can be buffered and backwards control information can be sent to the source of the message to delay any more packets until the displaced packet(s) held in storage can be sent. However, this requires each intermediate node to have enough memory to hold every packet of the message starting with the first displaced packet until the source node stops sending data. It also requires additional special control information to be sent (to slow or stop the message) every time any normal control information is sent, creating an avalanche effect.

One solution to this problem is to send the displaced packet out the link as soon as the control information has been sent without waiting for a break in the message. As the displaced packet is sent, the packet which follows it would have to be buffered and as that packet is sent the one following it would have to be buffered and so on until the message ends.

For this method to function effectively, two buffers would be required. One buffer would store the displaced packet while the control information was being sent. Then as the contents of the first buffer are being sent, a second buffer would be necessary to hold the next packet. Once the first packet is sent and the transmission of the next packet is started, the first buffer is freed for use in buffering the third packet after the delay.

If more control information needed to be sent before the end of the message, one more buffer would be required to effectively delay the remainder of the message by two packets. As long as there was a limit on the number of times control information could interrupt the message, one could supply enough buffers to guarantee that the system would always be able to delay any message enough to send all required control information.

The problem is that the amount of control information over a link is unrelated to the message traveling in the same direction over that same link. Therefore, predefining the memory requirements for a particular job is not possible and this method is not commercially practical.

The subject invention enhances the function of multi-hop communication networks used to transfer data between processing points in a distributed data processing system. An example will now be described using the following figures:

Figure **1** is a block diagram of a multiple node system;

Figure **2** is a block diagram of the nodal linking architecture used in the system of Figure 1;

Figure **3** is a block diagram of the inbound and outbound control logic used in the architecture of Figure 2;

Figure **4** is a flowchart of the outbound control logic; and

Figure **5** is a block diagram of the inbound control logic.

Consider the simple distributed data processing system and communication network shown in Figure **1**, consisting of a number of homogenous or heterogeneous data processing sites, called nodes (boxes **1, 2, 3, 4, 5, 6 and 7** represent nodes). Each node can be as simple as a small storage device (disk or memory) or as complex as a complete multiprocessing computer system (personal computer). It is, however, assumed that each node is able to both receive data blocks from other nodes and send data blocks to other nodes.

A distributed data processing system also consists of a communication network used to send blocks of data between nodes. One common type of communication network consists of a number of point to point transmission systems, called links, which interconnect the nodes in a regular or irregular pattern. For example, in Figure **1** lines **12, 14, 23, 25, 35, 45, 56 and 67** represent links.

This invention is applicable to links of any length, including links which connect two nodes in the same cabinet and links which connect two nodes in different cities. The type of transmission facility used is also not important as long as each link supports full-duplex transmission (data transfer in both directions at the same time).

For simplicity, assume that data sent between two nodes connected with a link is sent using that link. For example, in Figure **1** data sent from node **1** to node **2** would be sent across link **12**. When data is sent between two nodes which are not directly connected, multiple links will be used. For example, data sent from node **1** to node **6** might use links **12, 25 and 56.** This invention is applicable regardless of the mechanism used to find a path through the network and the mechanism used to handle the data within source, destination or intermediate nodes (nodes **2** and **5** in the previous example) used during the transmission of a data block.

Any type or format of data sent across the links can be used. In particular, there is no requirement that the data be split into separately routed packets (fixed size blocks of data) or that there be any restrictions on the size of data blocks sent. The term message will be used to refer to a data block being transferred between nodes.

Referring to Figure **1**, consider the transmission of a message from node **1** to node **6.** While the data is being sent from one node to another it may be necessary for the receiving node to send information back to the sending node in order to complete or control the data transmission. The following list gives some examples of the type of control information which may be sent.

1. When the message reaches the destination node (node **6** in the example), information may be sent back to the source node (node **1**) to ask for a re-transmission of the message if it was not received correctly.
2. If a link being used during the message transfer fails, then control information may be sent back to abort the message transfer without waiting for the end of the message.
3. If the destination node or any of the intermediate nodes (nodes **2** and **5**) are buffering the data as it is sent and those buffers fill up, information may be sent back to the sending node to temporarily pause the data transmission until the buffer overflow condition ends. At that point, another piece of control information may be sent backwards to restart the message.
4. If the choice of links which the message uses is determined dynamically at the beginning of the

message transfer then control information may be sent back by links under consideration to indicate whether or not a given link is available to be used to transfer the message.

If a given link (for example, link **12**) is actively sending a message in only one direction (node **1** to node **2**), then given that links are required to be full duplex, control information can be sent backwards by using the idle transmission capability normally reserved for sending messages in the opposite direction (node **2** to node **1**).

This simple technique does not work, however, if a link is busy sending messages in both directions. In our previous example, if one message is being sent from node **1** to node **2** and another message is being sent, at the same time, from node **2** to node **1** then there is no obvious way to send control information related to either message backwards across link **12**.

The problem is therefore how to send control information backwards across one or more links in a communication network when those links may already be busy sending unrelated messages. Based on the characteristics of the system used, the following requirements must be met:

1. The need to send control information in the system should not place any restrictions on the length of messages which can be sent in the network. Thus a message can vary in length.
2. Control information should be sent as soon as possible after it is generated. In particular, the sending of the control information can not be delayed until the end of any other message which may be blocking the transmission of the control information. This is because messages can be any length (see requirement 1) making the wait for an end of message undefined and potentially very long.
3. When a message is sent over multiple hops, there should no requirement for any significant amount of memory (more than a few bytes) in any of the intermediate nodes. This is especially important since messages can be any length making the amount of memory needed to store the longest messages at an intermediate node very large. This requirement implies that the network does not use store-and-forward techniques where each intermediate node stores each message in local memory before forwarding it on to the next link towards the destination node.
4. If control information has to be sent over a link, no other links actively sending that message or the source node generating that message should have to be explicitly told to slow down in order to provide room for the control information.
This is because the control information which would have to be sent to slow down a message would itself require another message to slow down as it is sent backwards over a link. This could cause an avalanche effect eventually flooding the network with requests to slow down.
5. If control information is sent over a link which is busy sending a message, then the control information will steal bandwidth from the message. However, in the absence of control information, that message should use the full bandwidth of the link. Specifically, none of the available bandwidth of a link can be reserved for control information which may not be sent.
6. Although it can be assumed that there is a limit to the amount of control information sent backwards for any given message, because of requirement 1, there is no relationship between any given message being sent over a link and the control information sharing that link.
In order words, a given link may have one long message flowing from left to right and lots of short messages flowing from right to left. Because the control information associated with a short message flows backwards control information might have to be sent left to right over the link before the message which is flowing left to right ends.
7. If control information is sent over a link which is busy sending a message, then the message transfer may be interrupted locally but the data in that message must remain in order with no data lost and the message can not be prematurely terminated.

For simplicity, it will be assumed that every link in the network has the same bandwidth and the full bandwidth of every link is used when sending messages. This means that if a message is sent between two nodes which are not directly connected (for example, node **1** to node **6**), the message will use the entire bandwidth of every intermediate link in the path (links **12, 25 and 56** in our example).

If this assumption were relaxed such that any of the intermediate links, except for the first link, had more bandwidth than the first link, then when a message was sent over that path, it would only use a subset of the of the bandwidth available over each of the faster intermediate links. The extra bandwidth in those intermediate links could then be used to send control information which would reduce the applicability of this invention to those slower links in the network which were completely used.

The situation gets a little more confusing if the first link has more bandwidth than any other link in a multi-link path. In that case, a message can either be sent out using only part of the bandwidth of the first link to avoid overloading the other links. Or, the full bandwidth of the first link could be used which would cause one of the intermediate nodes to receive more data than it could send. In the second case, the intermediate node would be required to buffer the extra data to prevent it from being lost. This, in turn, requires that intermediate node to have significant amounts of buffer memory in violation of requirement 3 (above).

Since the invention is most beneficial when links in the system are running at full bandwidth, it will be assumed that every link is running at the same speed equal to the full bandwidth of any link to simplify the description. In networks where link speed varies, the invention is mainly useful for those links which are used at full bandwidth.

Using a technique where idle links are BLOCKED as if they were busy limits the amount of control information which could delay any given message and guarantees that a small amount of buffer memory is sufficient. A simple, yet effective, way of delaying a message so control information can be inserted is employed.

Consider a single full-duplex link **12** between two nodes **1** and **2** in a communication network as shown in Figure **2**. Messages sent forward from node **1** to node **2** will be generated within node **1** and given to logic block **90** to be sent over the link **12**. Likewise, messages sent from node **2** to node **1** will, inside node **1**, be extracted from link **12** by logic block **91** and given to node **1** for processing or forwarding through other links. Control information associated with any message flowing from node **1** to node **2** will be sent backwards from node **2** to node **1** by having the control information inserted into any message stream flowing from node **2** to node **1** by node **2**'s outbound link logic **93**. The control information is then separated from that message stream by node **1**'s inbound link logic **91**.

The logic within a node which sends, receives and routes messages between links as well as the logic which processes control information is beyond the scope of this invention. What is described by this invention is the processing of message and control information between the routing logic within the node and the link level drivers and receivers. This logic, which is shown as blocks **90** and **91** in Figure **2** is detailed in Figure **3**.

Forward flowing messages are logically divided into packets, each of which is the same size as the smallest piece of control information. Thus, packets are typically between one to four bytes. The stream of packets from the node are put into a first-in-first-out buffer or FIFO **100**. The output of the FIFO is fed into multiplexor **101** where either the message data, backwards control information, or special MSG-BLOCK or BLANK characters can be selected. The output of the multiplexor is then fed into the link driver **102** as depicted in Figure **4**.

In the absence of any backwards control information, message data is sent through the FIFO without buffering, fed through the multiplexor and sent over the link with a minimum of delay. The bandwidth of the link should be designed to match the speed of message data coming from the node.

When backwards control information is presented to be sent over the link, the outbound control logic **103**, fully detailed in the flowchart depicted in Figure **4**, is designed to switch the multiplexor **101** to allow the control information access to the link. During that time, the message is delayed by allowing one or more packets to back up in the FIFO. After the control information is sent, the multiplexor is restored and the message continues where it left off.

If there is neither message data nor backward control characters to be sent over the link, then the outbound control logic will switch the multiplex to its own output and sends a special BLANK control packet over the link to indicate that there is a gap in the message.

Every driver is connected to a receiver in another node. The receiver in this node which connects to the driver at the other end of the link is shown in Figure **3** as **104**. Data from the receiver is fed into a de-multiplexor **105** where message data is separated from control information.

The inbound control logic **106**, fully detailed in the flowchart depicted in Figure **5**, is designed to remove both BLANK and MSG-BLOCK packets from the inbound data stream. These two characters look like backwards control character, except that they are generated by the outbound control logic rather than the node. All data and control character except for BLANK and MSG-BLOCK are sent to the node for futher processing.

The method used to distinguish message data from control information depends on the type of link used and is beyond the scope of this invention. In some systems, an escape character is used to mark the beginning of control information. In other systems, the set of unique packets (characters) which can be sent over the link is larger than the set needed for message data. The packet types unused for message data can be used for control information.

The width of the FIFO is usually set to the packet size. When a message is delayed in order to insert control information, it is the FIFO which acts like memory to hold the packets. Thus, the depth of the FIFO determines how much control information can be sent without dropping packets.

The FIFO should be designed with a relatively short fall through time relatative to a packet transmission time. The fall through time is defined as the time from when data enters the FIFO until it can be read from the FIFO. As long as the fall through time is relatively short, it can be assumed that when data is not available from the FIFO that the FIFO is mostly empty.

In order to prevent the FIFO from overflowing, the outbound control logic, **103** in Figure **3**, monitors the amount of room remaining in the FIFO as fully detailed in the flowchart depicted in Figure **4**. When the FIFO is almost full (defined later) then the outbound control logic will interrupt the message to send its own piece of control information, called a MSG-BLOCK.

The MSG-BLOCK will be sent over the link to the receiver at the other end of the node. This piece of control information will then be stripped from data stream by the inbound control logic **106** in the other node, as fully detailed in the flowchart depicted in Figure **5**. When a MSG-BLOCK is received, the inbound control logic in the receiving node sets a control line called BLOCKED which is monitored by the node. This BLOCKED signal does not effect any message currently being sent through the node; however, after the current message finished, no new messages can be accepted by that link until the BLOCKED signal is cleared.

After a MSG-BLOCK is detected by the inbound control logic **106**, that control logic will start to look for a BLANK control packet as shown in function block **235** of Figure **5**. When a BLANK is detected, the inbound control logic can assume that the FIFO at the other end of the link is almost empty. At this point, the BLOCKED signal can be reset as shown in function block **240** of Figure **5**. This usually occurs at the end of the message, although a break in the middle of the message is possible if the message is being transmitted slower that the bandwidth of the link.

To fully comprehend the above, it is necessary to review what can be assumed about the number of backwards control packets which may be seen.

Referring to Figure **1**, assume a message A is being sent from node **1** to node **6** using links **12, 25 and 56.** At the same time a message B is being sent from node **2** to node **4** using links **12 and 14.** Notice that link **12** has two messages going across it, one in each direction.

Each message can cause some amount of backwards control information to be generated. Let M equal the maximum amount of control information generated in response to any given message. Notice that M is unrelated to the amount of control information which can interrupt any given message. In our example, if nine short messages (B1 - B9) are sent from node **2** to node **4** at the same time that one long message A is sent from node **1** to node **6** then up to 9*M pieces of control information may have to be inserted into message A as it is sent over link **12** from node **1** to node **2.**

Assume that the FIFO in node **1** has room for 3*M packets. If, during the transmission of message A, more than 3*M pieces of control information are sent across link **12**, then some packets of message A will be lost. To avoid losing parts of message A, it is necessary to limit the amount of control information generated by limiting the number of messages (B1 - B9) which flow in the opposite direction.

With reference to Figure **2**, message A is flowing from node **1** through block **90** across link **12** through block **92** and into node **2**. Once the message has entered into node **2** it will be routed across link **25** which is not shown. Message B is flowing from node **2** through block **93** across link **12** through block **91** and into node **1** enroute to node **4.**

Every message B which flows from node **2** to node **4** will generate control information flowing from node **4** to node **2.** The control information will be sent into node **1** through block **90** across link **12** through block **92** and into node **2,** stealing bandwidth from message A which is using the same link **12.** Every time a piece of control information is sent across link **12,** another packet of message A is delayed and the FIFO **100** in Figure **3**, inside block **90** fills a little more.

When the number of empty stages remaining in the FIFO **100** inside block **90** is equal to 2+M, the outbound control logic **103** inside block **90** will cause a MSG-BLOCK to be sent across link **12** to node **2.** The MSG-BLOCK transmission delays message A some more and uses up one more empty stage of the FIFO.

The MSG-BLOCK character is removed from the link by the inbound control logic **106** inside block **92.** That control logic then tells node **2** that link **12** is BLOCKED. Assume that a message B3 is being sent from node **2** to node **1** when the BLOCKED signal is set. The BLOCKED state does not effect message B3 because that message is already in progress. In addition, message B3 is still free to cause up to M more pieces of control information to be sent across link **12** because the FIFO has M + 1 more empty stages.

After message B3 has ended, any new message B4 will not be allowed to start being sent across link **12** as long as link **12** is flagged as BLOCKED. In an adaptive routing system, the routing logic inside node **2** is free to select another path for the message if possible. Otherwise, messages from node **2** to node **4** will be delayed or rejected until the BLOCKED condition is cleared as if the link was busy with another message.

Meanwhile the inbound control logic **106** in block **92** is monitoring the message A data coming across link **12** from node **1.** If a BLANK control packet is detected, it implies that the FIFO **100** inside block **90** is either empty or almost empty. When this event occurs, the inbound control logic resets the BLOCKED condition and message B traffic is again allowed to flow from node **2** to node **1.**

Because bandwidth is stolen from message traffic in order to send control information, the net result is that each message from which bandwidth is taken is stretched out so that it takes longer to send. Thus there must be an idle period between messages into which the stretched messages can be extended.

Because there is no explicit flow control information required by this invention, the source node for each message remains unaffected by the insertion of control information. The destination node, however, sees messages which have been interrupted by control information arriving at a slower rate than they were sent. In a system where every message is acknowledged (by sending control information back to the source node) before another message is sent, this will have a net effect of delaying the acknowledgement to the source node. On the other hand, the delay in the acknowledgement is proportional to the amount that the message was delayed and acts as a convenient flow control mechanism to ensure enough idle time between messages.

Because the size of every FIFO is much smaller than the message size, some links will occasionally get BLOCKED. The percent of links which are BLOCKED at any given time is related to the type and amount of message traffic as well as the size of each FIFO. Thus the size of each FIFO should be designed to trade off the loss of network bandwidth due to link BLOCKED conditions for real estate available on each node for FIFO memory.

The minimum size for a FIFO is 3+M packets (where M is the maximum number of backwards control characters which can be generated in response to any given message). In practice, FIFO sizes on the order of 3+3*M have been shown to give good performance (less than 1% links BLOCKED during peak message traffic).

An important characteristic of this invention is that long messages sent over multiple hops do not tend to cause congestion over their entire route. This tends to limit the number of BLOCKED links any given message can cause no matter how many links that message is sent over.

To understand this, consider a message A being sent from node **1** to node **6** over links **12, 25 and 56.** If backwards control information is sent over link **12** from node **1** to node **2**, message A will be delayed in the FIFO inside node **1**. In addition, the stream of message A packets flowing through node **2** will have a gap due to the removed control information. This gap allows any downstream FIFOs (in node **2** or node **5**) to drain a little. Thus, the insertion of control information early in the message actually relieves any backup later in the message reducing the possibility that other links will have to be BLOCKED.

This invention is used in the routing system of a multi-dimensional nodal system to optimize the transmission of control information in addition to messages between nodes. Nodes are connected using pairs of serial links (one in each direction). The links vary in length from one to twenty feet. Each link supports two-hundred and sixty-eight unique characters (encoded in ten bits) of which two hundred and fifty-six are used for message data and the remaining twelve are used for control information. Thus the packet size and the corresponding width of every FIFO is set to one character (10 bits).

Nodes are interconnected in a four dimensional torus network with an additional link at every node for I/O. Every node has nine links, each of which contains a FIFO and outbound and inbound control logic as depicted in Figure **3**. The FIFO size has been set to sixteen packets because every message can generate up to four backwards control packets.

Inside the node, data is routed between links using the "Adaptive Routing in a Parallel Computer System" (EPA410568). The adaptive routing system handles the BLOCKED condition as if the link were busy with another message and, in general, has no problem routing around any BLOCKED links (the LINK-CLOSE and ACKNOWLEDGE commands therein correspond to backwards control information described herein).

Claims

1. Network communication apparatus for transmitting selected control information and message data between a plurality of nodes, comprising:
   (a) communication means (104) for transmitting message data;
   (b) memory means (100) for storing portions of the message data;
   (c) processor means (103) for selecting appropriate control information to transmit based on current communication status; and
   (d) multiplexor means (101) for transferring selected control information or message data to the communication means for transmission to a subsequent node.

2. Apparatus as recited in claim 1, further comprising transmitting means for full-duplex communication.

3. Apparatus as recited in any preceding claim, wherein the control information includes information requesting a re-transmission of the information.

4. Apparatus as recited in any preceding claim, wherein the control information includes information requesting a transmission abort.

5. Apparatus as recited in any preceding claim, wherein the control information includes information requesting a temporary pause of further transmission of information.

6. Apparatus as recited in any preceding claim, wherein the control information includes information requesting a restart of transmission of information.

7. Apparatus as recited in any preceding claim, wherein the control information includes status information on transmission between nodes.

8. Apparatus as recited in any preceding claim, wherein the transmission speed remains constant

during transmission of the control information.

9. A method for communication in a network comprising a plurality of nodes, the nodes comprising processor means (103), memory means (100), multiplexor means (101) and communication means (102), whereby selected control information and message data is transmitted between nodes, said method comprising the steps of:
   (a) storing a portion of message data in said memory means;
   (b) selecting appropriate control information to transmit based on current communication status; and
   (c) transferring selected control information or message data via the multiplexor means to the communication means for transmission to a subsequent node.

10. A method as recited in claim 9, further comprising the step of transmitting information in packets.

11. A method as recited in claim 10 or 11, further comprising the step of receiving message data and sending the control information synchronously.

12. A method as recited in any of claims 9 to 11, further comprising the step of transmitting variable length information.

13. A method as recited in any of claims 9 to 12, further comprising the step of transmitting the control information and the message at variable rates at some of the plurality of nodes.

14. A method as recited in any of claims 9 to 13, wherein appropriate control information is selected from a set of information stored in said memory means.

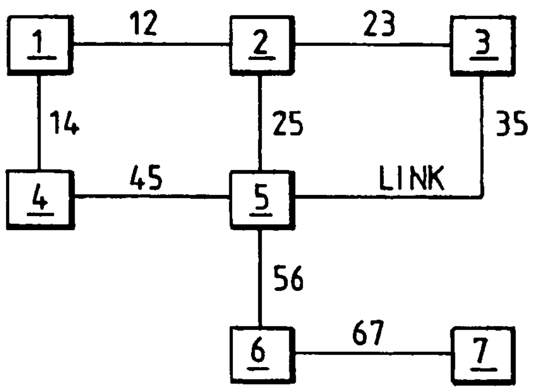
1
2
3
4
5
6
7
12
23
14
25
35
45
LINK
56
67

FIG. 1

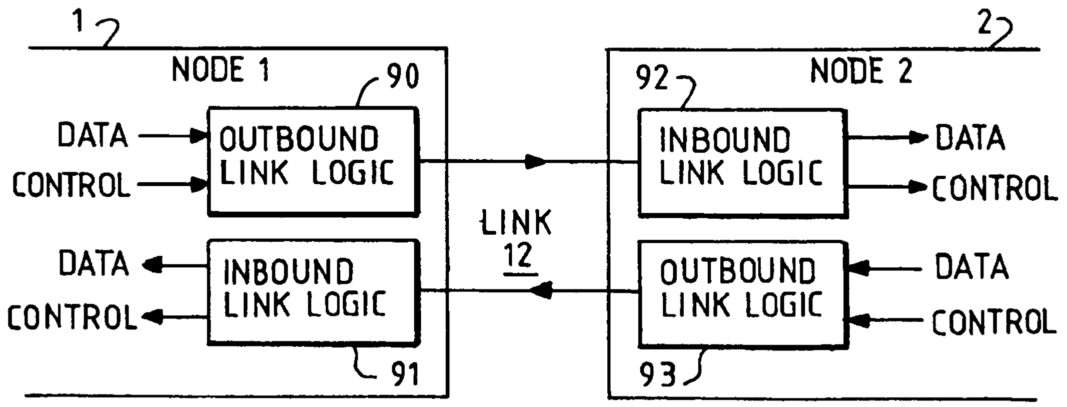
1
NODE 1
90
DATA
CONTROL
OUTBOUND LINK LOGIC
DATA
CONTROL
INBOUND LINK LOGIC
91
LINK
12
2
NODE 2
92
INBOUND LINK LOGIC
DATA
CONTROL
OUTBOUND LINK LOGIC
DATA
CONTROL
93

FIG. 2

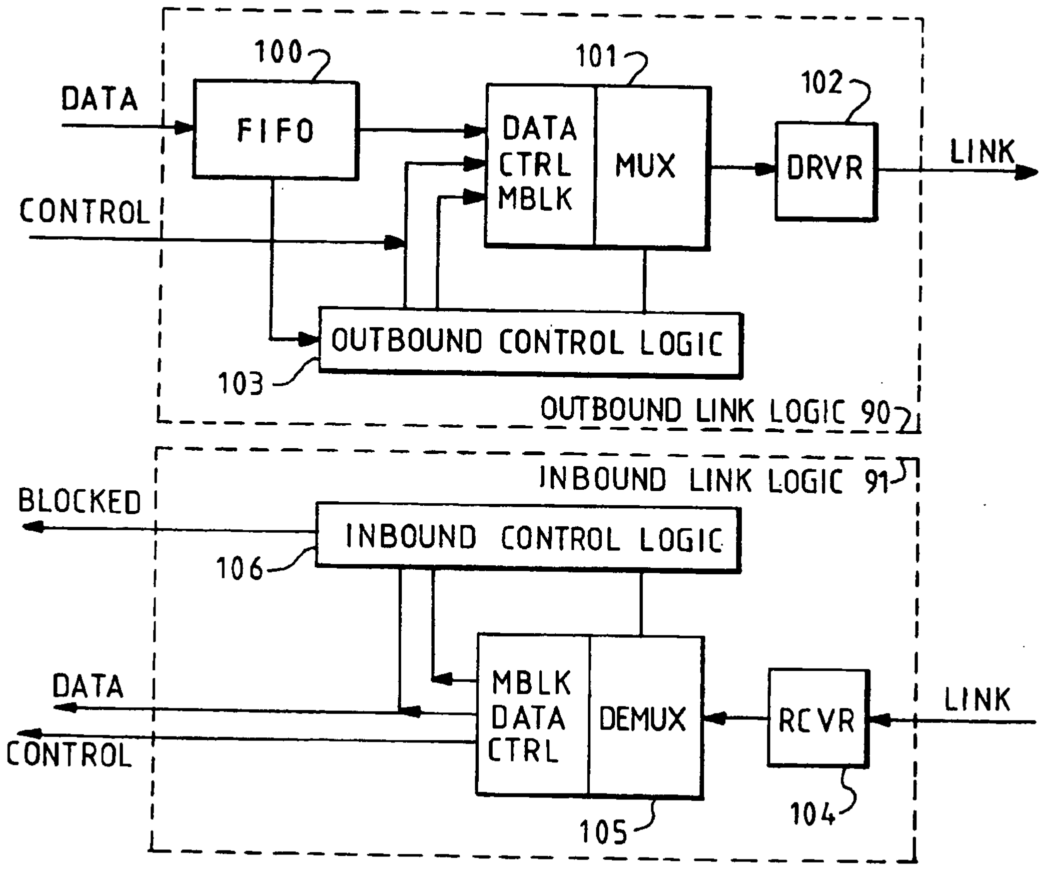
DATA
CONTROL
100
FIFO
101
DATA
CTRL
MBLK
MUX
102
DRVR
LINK
OUTBOUND CONTROL LOGIC
103
OUTBOUND LINK LOGIC 90
INBOUND LINK LOGIC 91
BLOCKED
INBOUND CONTROL LOGIC
106
DATA
CONTROL
MBLK
DATA
CTRL
DEMUX
RCVR
LINK
105
104

FIG. 3

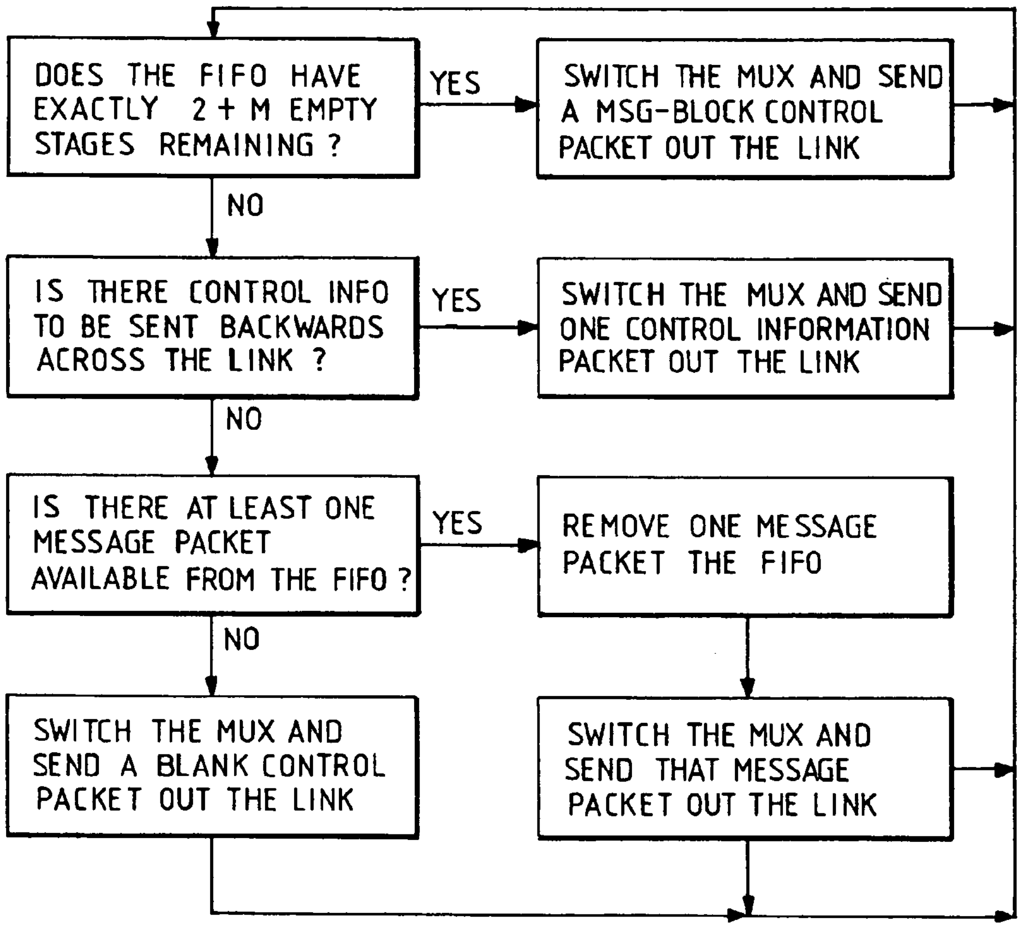
DOES THE FIFO HAVE EXACTLY 2 + M EMPTY STAGES REMAINING ?
YES
SWITCH THE MUX AND SEND A MSG-BLOCK CONTROL PACKET OUT THE LINK
NO
IS THERE CONTROL INFO TO BE SENT BACKWARDS ACROSS THE LINK ?
YES
SWITCH THE MUX AND SEND ONE CONTROL INFORMATION PACKET OUT THE LINK
NO
IS THERE AT LEAST ONE MESSAGE PACKET AVAILABLE FROM THE FIFO ?
YES
REMOVE ONE MESSAGE PACKET THE FIFO
NO
SWITCH THE MUX AND SEND A BLANK CONTROL PACKET OUT THE LINK
SWITCH THE MUX AND SEND THAT MESSAGE PACKET OUT THE LINK

FIG. 4

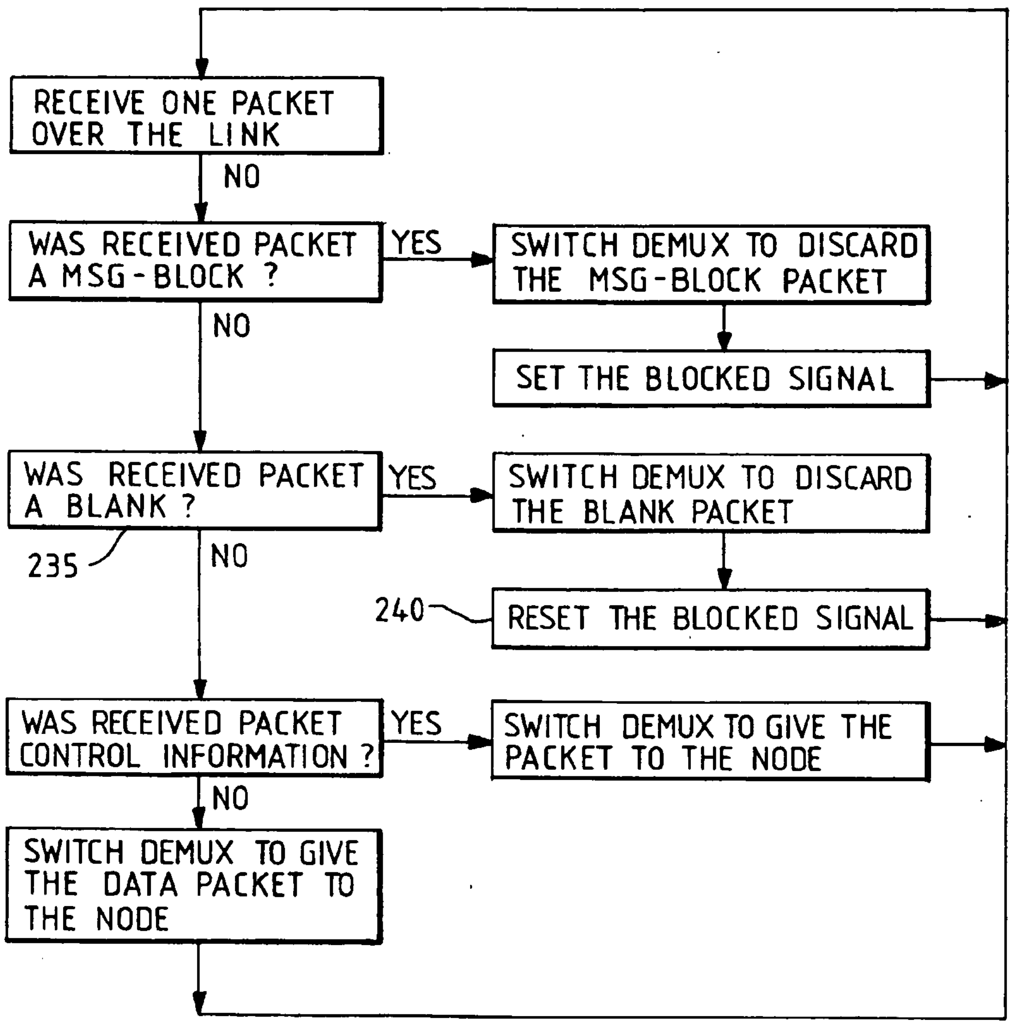
RECEIVE ONE PACKET OVER THE LINK
NO
WAS RECEIVED PACKET A MSG-BLOCK ?
YES
SWITCH DEMUX TO DISCARD THE MSG-BLOCK PACKET
SET THE BLOCKED SIGNAL
NO
WAS RECEIVED PACKET A BLANK ?
YES
SWITCH DEMUX TO DISCARD THE BLANK PACKET
235
NO
240
RESET THE BLOCKED SIGNAL
WAS RECEIVED PACKET CONTROL INFORMATION ?
YES
SWITCH DEMUX TO GIVE THE PACKET TO THE NODE
NO
SWITCH DEMUX TO GIVE THE DATA PACKET TO THE NODE

FIG. 5